NoSQL

SQL vs NoSQL

- RDBMS (Relational database management systems)
  - Relational model
  - SQL
  - Provide strong data integrity
  - ACID
    - Atomicity – Atomic transactions
    - Consistency – Data integrity and consistency before and after transaction
    - Isolation – isolated transactions, concurrent transactions do not interferre with each other
    - Durability – When a transaction is committed, its effects are permanent (event after hardware / system failure)
  - Suitable for applications requiring complex queries, multi-row transactions, and strong data integrity, such as financial systems and enterprise applications.

SQL vs NoSQL

- NoSQL – „Not only SQL“
- Supporting all the ACID requirements prevents easy scaling of the database across multiple servers (nodes).
- Sometimes we would like to have higher transaction throughput and better scalability options at the cost of not fully adhering to ACID requirements.
- RDBMS – mostly vertical scaling
  - if we want to process more transactions, we need to use stronger servers (more RAM, more CPU, faster drives)
- NoSQL – mostly designed for horizontal scaling
  - If we need to process more transactions, we just add more servers

SQL vs NoSQL

BASE (vs ACID)

- **Basically Available:** high availability even if network or hadware failues (multiple nodes share the same data)
- **Soft state:** The system can be in a "soft" or intermediate state, which means that data consistency is not guaranteed at all times.
- **Eventual Consistency:** After a certain period of time, all nodes in a distributed system will hold the same data, assuming no new updates are made during that period.
  - If you read data immediately after writing it, you might not get the most recent update.
  - The system will eventually converge to a consistent state where all replicas have the same data.

SQL vs NoSQL

	SQL / RDBMS	**NoSQL**
integrity	is mission-critical	OK as long as most data is correct
data format	consistent, well-defined	unknown or inconsistent
data	is of long-term value	is expected to be replaced
growth	predictable, linear growth	unpredictable growth (exponential?)
querying	non-programmers writing queries	only programmers writing queries
fault tolerance	regular backups	automatic data replication among multiple nodes
distribution	access through master server	data sharding (partitioning), multiple nodes with the same data

SQL vs NoSQL

- RDBMS:
  - SQL language
  - Relational model
  - ACID
  - difficult to scale horizontally but:
    - Distributed RDBMS exists (Google spanner, CockroachDB)
- NoSQL:
  - Not using the relational model (nor the SQL language)
  - No / flexible schema - fields can be freely added to any record
  - BASE
  - Designed to run on many nodes (horizontally scalable)

Basic types of NoSQL

- Key-value stores
- Document databases
- Column-family stores
- Graph databases

Key-value stores

- Simple hash table
- Used when all accesses to the DB are via primary key
- Used for
  - web sessions,
  - user profiles and preferences
- Memcached, MapDB, LevelDb, Redis


```
{
 "session_12345": {
  „logged_user_id": "001",
  "login_time": "2024-12-08T15:45:00Z",
  "last_activity": "2024-12-08T16:15:00Z",
  "cart_items": [
   {
    "product_id": "A1001",
    "quantity": 2
   },
   {
    "product_id": "A2002",
    "quantity": 1
   }
  ],
},
 "session_67890": {
  "user_id": "002",
  "login_time": "2024-12-08T16:00:00Z",
  "last_activity": "2024-12-08T16:30:00Z",
  "cart_items": [
   {
    "product_id": "B3003",
    "quantity": 3
   }
  ],
 }
}
```

Document databases

- Hierarchical tree data structures
- Nested associative arrays (maps)
- JSON / BSON / XML

- MongoDB

```
Users: [
{
      "user_id": "001",
      "name": "Alice Smith",
      "email": "alice.smith@example.com",
      "signup_date": "2024-01-15T08:30:00Z",
      "preferences": {
        "newsletter": true,
        "notifications": ["email", "sms"]
      },
      "friends": ["002", "003"]
    },
  {
      "user_id": "002",
      "name": "Bob Johnson",
      "email": "bob.johnson@example.com",
      "signup_date": "2024-02-20T09:00:00Z",
      "preferences": {
         "notifications": ["email"]
      },
      "friends": ["001"],
      "address": {
         "street": „Greenfield 10",
         „city": „Flowersburg",
      },
    }
]
```

MongoDB - example

SQL: SELECT * FROM users

MongoDB: db.users.find()

SQL: SELECT * FROM users WHERE user_id = "3"

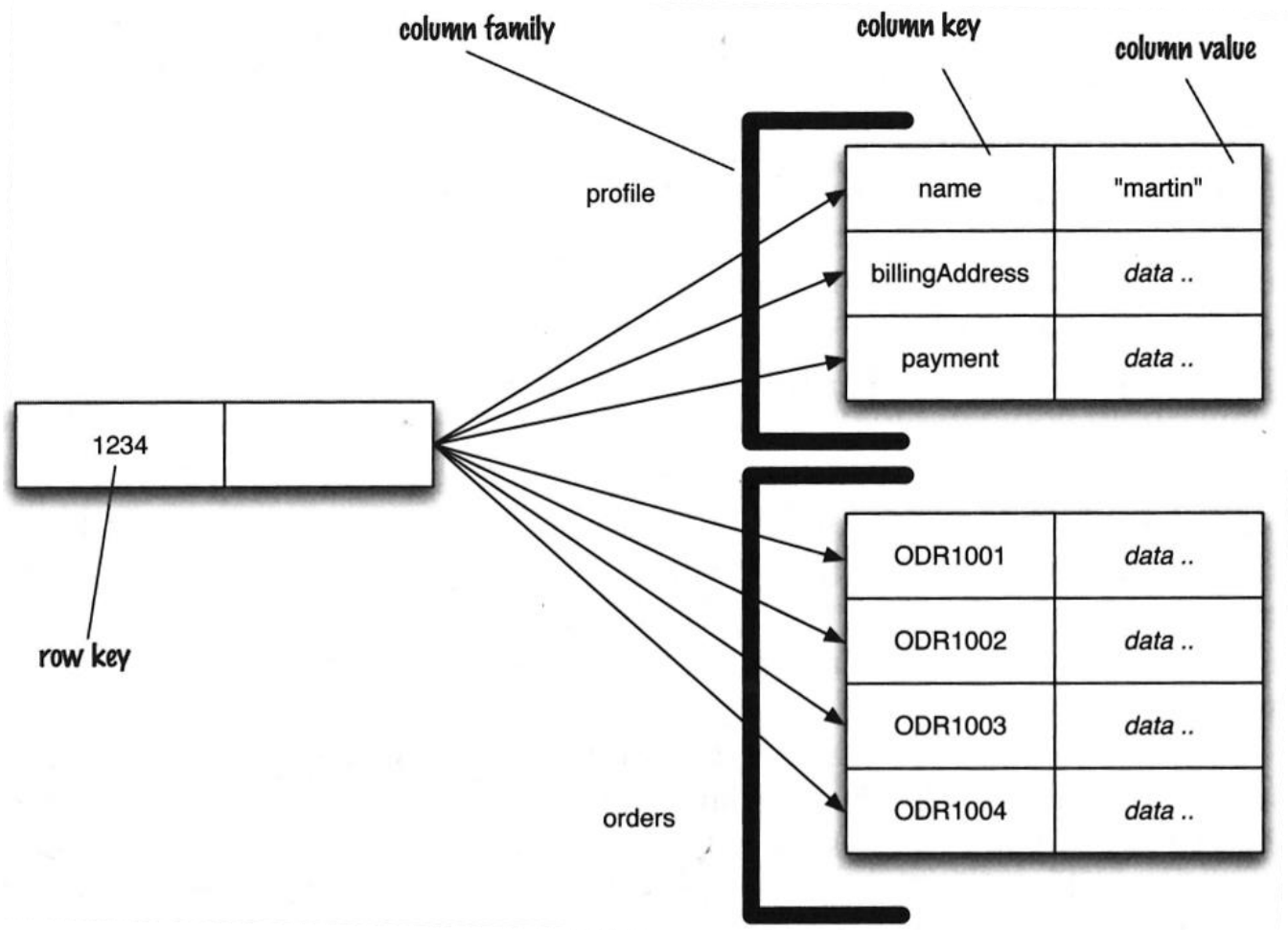
MongoDB: db.users.find({„user_id":"3"})

SQL: SELECT firstname,lastname FROM users WHERE user_id=5

MongoDB: db.users.find({„user_id":"5"},{firstname:1,lastname:1})

Column-family stores

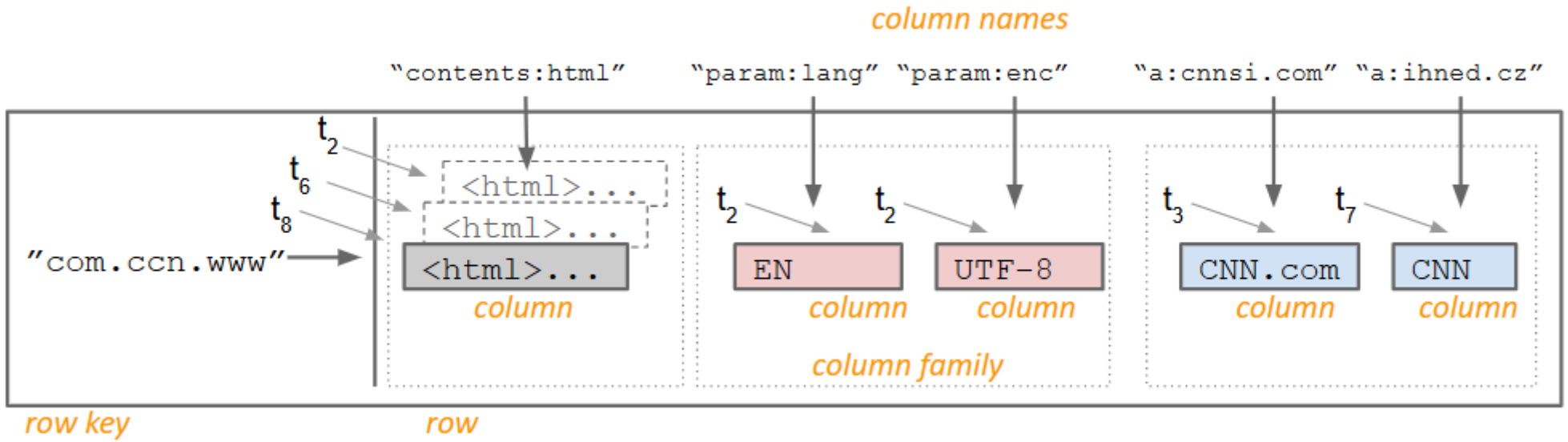
- Rows that have many columns associated with a row key
- Column families are groups of related data (columns) that are often accessed together
- Apache Cassandra



source: Sadalage & Fowler: NoSQL Distilled, 2012

Google BigTable (2008)

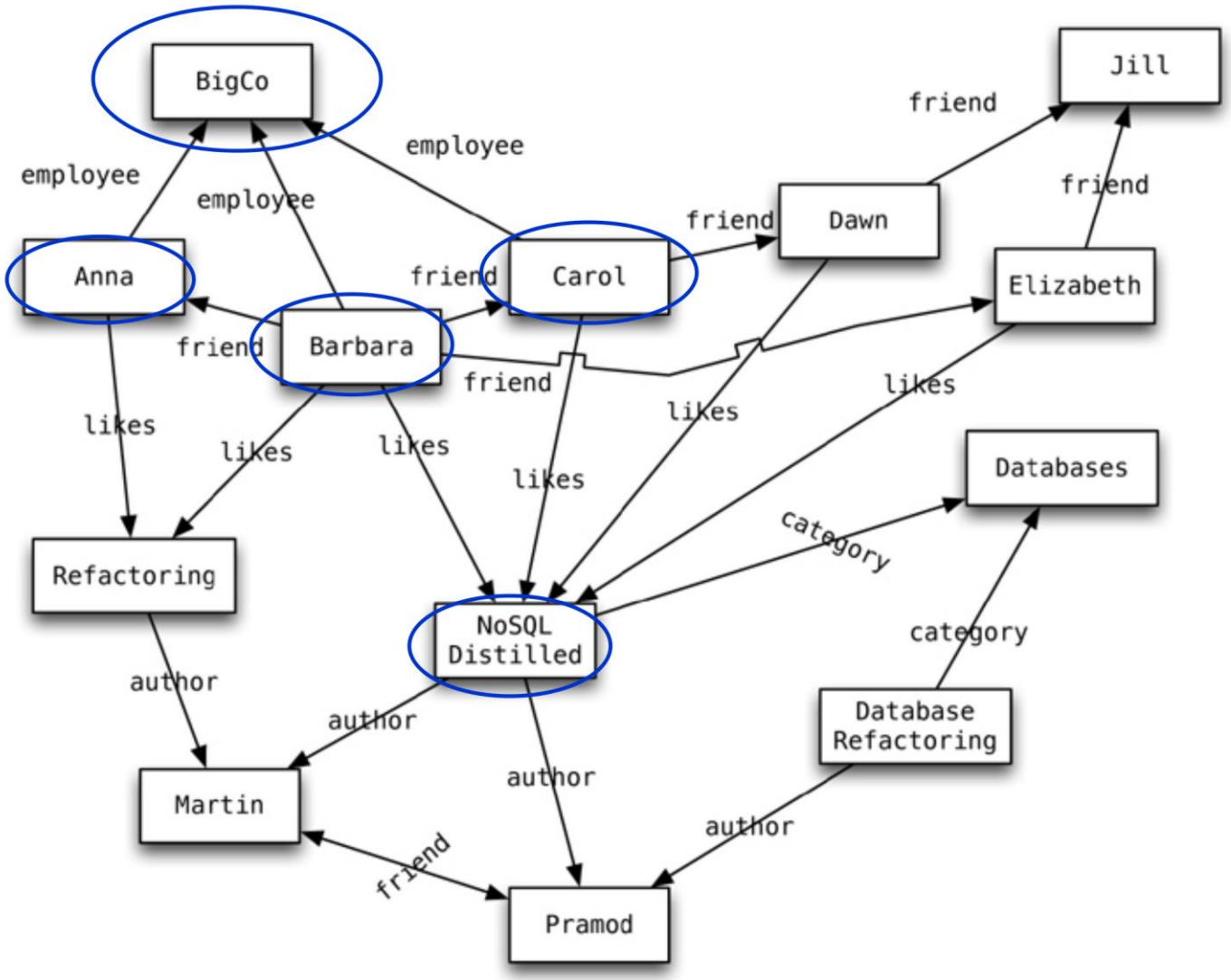
BigTable = sparse, distributed, persistent, multi-dimensional sorted map indexed by *(row_key, column_key, timestamp)*



https://disa.fi.muni.cz/david-novak/teaching/pa195-nosql-databases/lectures/

Graph databases

- Graph nodes are objects
  - Each node can have properties (name, address...)
- Edges have directional significance
  - Edges have types (like, employee,...)



Resources

- https://disa.fi.muni.cz/david-novak/teaching/pa195-nosql-databases/lectures/

- https://onecompiler.com/mongodb